НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки’’

на тему: Динамічне середовище геометричного моделювання для мобільних телефонів

Виконав: студент 4 курсу, групи ТР-51
Загребельний Євгеній Олегович _____
(прізвище, ім’я, по батькові) (підпис)
Керівник д.т.н., професор Аушева Н.М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)
Рецензент к.т.н., ст. викладач Кондратюк В.А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки’’

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.В. Коваль

(підпис)

”___”_____2019 р.

ЗАВДАННЯ

на дипломну роботу студенту

Загребельному Євгенію Олеговичу

(прізвище, ім’я, по батькові)

1. Тема роботи “Динамічне середовище геометричного моделювання для мобільних телефонів”
керівник роботи д.т.н., професор Аушева Наталія Миколаївна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ” 22 ” травня 2019 р. № 1325 - с

2. Строк подання студентом роботи 10 червня 2019 р.
3. Вихідні дані до роботи текстовий файл з розширенням .txt, в якому зберігається поверхня у проміжному етапі
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі методи та засоби для створення динамічного середовища геометричного моделювання, обрати методи та засоби для створення динамічного середовища геометричного моделювання на мобільному телефоні, розробити структуру та діаграму класів, розробити алгоритм для керування динамічною поверхнею Безьє, реалізувати програмне забезпечення.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень)
1. Постановка задачі 2. Аналіз існуючих рішень для розвязку поставленої задачі 3. Теоретичне підгрунття методів розвязання поставленої задачі 4. Опис розробленого програмного забезпечення 5. Методика роботи користувача з програмною системою

Дата видачі завдання " 10 " жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	20.10.2018-29.12.2018	
2.	Розробка архітектури та загальної структури системи	7.02.2019-10.03.2019	
3.	Підготовка матеріалів	10.03.2019-13.04.2019	
4.	Програмна реалізація системи	20.04.2019-13.05.2019	
5.	Захист програмного продукту	15.05.2019	
6.	Оформлення пояснювальної записки	16.05.2019-5.06.2019	
7.	Передзахист	28.05.2019	
8.	Захист	17.06.2019-22.06.2019	

Студент _____ Загребельний Є.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Аушева Н.М.
(підпис) (прізвище та ініціали)

ЗМІСТ

Вступ 8

1Постановка задачі РЕАЛІЗАЦІЇ ДИНАМІЧНОГО СЕРЕДОВИЩА ГЕОМЕТРИЧНОГО МОДЕЛЮВАННЯ ДЛЯ МОБІЛЬНИХ ТЕЛЕФОНІВ 10

2Аналіз існуючих методів та програмних забезпечень для створення Динамічного середовища Геометричного моделювання 11

2.1Програмне забезпечення “AutoCAD” 12

2.2Програмне забезпечення “Unigraphics” 13

3Аналіз існуючих рішень геометричного моделювання 14

3.1Технології геометричного моделювання 14

3.2Методи моделювання кривих 15

3.2.1Базові поняття для створення кривих 15

3.2.2Основні поняття про криву 16

3.3Моделювання поверхні Безьє 16

3.3.1Крива Безьє 16

3.3.2Властивості поверхні Безьє 17

3.4Побудова поверхні Безьє 19

3.5Двомірні геометричні перетворення 20

3.5.1Трансляція 20

3.5.2Обертання об’єкту 21

3.6Тривимірні геометричні перетворення 22

3.6.1Тривимірна трансляція 22

3.6.2Тривимірне обертання 23

4Опис розробленого програмного забезпечення 25

4.1Середовище розробки Visual Studio 2017 25

4.2Ігровий рушій Unity3D 27

4.3Система контролю версій 28

4.4Бібліотека класів UnityEngine 29

4.5Архітектура програми 30

4.6Діаграма класів 30
4.7Програмний модуль побудови поверхні Безьє і роботи з цією поверхнею. 31
4.8Програмний модуль роботи з камерою 32
4.9Програмні модулі роботи з файловою системою 32
5Методика роботи користувача з програмною системою 33
5.1Інсталяція та системні вимоги 33
5.2Сценарії роботи користувача з системою 36
5.3Деінсталяція програмної системи 38
висновки 41
Додаток А 44
Додаток Б 46
Додаток В 53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Unity3D — платформа для розробки гри (ігровий двигун);

C# script — сценарій написаний на мові програмування C#;

Android — операційна система для smartphone;

iOS — операційна система для smartphone;

Microsoft Visual Studio — середовище для розробки програмного забезпечення.

ВСТУП

В останні роки розвиток науки багато в чому пов’язаний з дослідженням динамічних процесів у деформації тіл. Мобільне графічне моделювання, як і комп’ютерне, являється підкласом інформаційного моделювання і являє собою моделювання об’єктів засобами комп’ютерної графіки. Графічна модель, як і будь-яка інша, описує реальний об’єкт лише з деяким ступенем наближення до дійсності. Динамічне графічне моделювання відтворює процес функціонування й розвитку об’єктів у часі й у просторі.

Динамічні графічні моделі дозволяють вивчати не тільки готові результати, але й розглядати процес їхнього одержання, дослідження, формують в користувача здатність продукувати нестандартні ідеї й рішення, сприяють розвитку інтуїції.

Під навчальною динамічною графічною моделлю будемо розуміти подання об’єкта або процесу предметної області з використанням різних видів інформації з метою вивчення й відтворення певних властивостей об’єкта, що здатне виконувати когнітивну функцію.

До динамічних графічних моделей висуваються наступні вимоги:

1) мультимедійність — в динамічних графічних моделях використовуються такі елементи мультимедійних технологій, як графіка, анімація, звук, текст, відео, чим забезпечується різноманітність форм подання інформації, адже інформація, що доводиться до користувача одночасно декількома каналами, сприймається більш ефективно;

2) інтерактивність — найкращою формою подання матеріалу є така, за якої кожний об’єкт на екрані доступний для вивчення, видозміни та комбінування з іншими об’єктами, що дозволяє користувачу зайняти позицію активного учасника навчального процесу, обирати індивідуальний темп і траєкторію навчання;

3) універсальність — можливість проілюструвати практично кожний елемент досліджуваних процесів і явищ, що у звичайних умовах зробити не можна;

4) компактність — файли моделей, створені засобами високорівневої мови програмування, мають малий розмір, що відіграє істотну роль при передаванні по мережі, а також при розміщенні на сервері.

Використання інтерактивних графічних моделей відіграє важливу роль в навчанні, тому що вони є важливим засобом, який дозволяє найбільш повно передати інформацію суб'єкту процесу навчання, до того ж з їхньою допомогою відбувається особиста участь користувача в одержанні інформації.

Ефективне використання можливостей мобільного моделювання для реалізації динамічних моделей у значній мірі залежить від програмного забезпечення, яке повністю автоматизує процес формулювання та розв'язання задачі.

Технології розвиваються з великою швидкістю. Кожного дня з'являються на світ нові моделі рішення, мови програмування, системи і багато іншого. А найцінніший ресурс, завжди був час, тому всі нові технології намагаються одночасно виходити і на телефони і на комп'ютери.

Більшість людей стикаються з нехваткою часу для включення комп'ютера, але телефон у них завжди при собі і завжди працює, тому перехід нових програмних забезпечень на телефон, кожного дня збільшується.

Записка містить 5 розділів.

У першому розділі описується постановка задачі реалізації динамічного середовища геометричного моделювання для мобільних телефонів.

У другому розділі проводиться аналіз існуючих методів та програмних забезпечень геометричного моделювання.

У третьому розділі описуються обрані методи та програмні забезпечення.

У четвертому розділі описується розроблена структура, діаграма класів, та алгоритм для керування динамічною поверхнею Безьє.

У п'ятому розділі описано інструкцію по роботі з програмним продуктом, системні вимоги та продемонстрований його інтерфейс.

1 ПОСТАНОВКА ЗАДАЧІ РЕАЛІЗАЦІЇ ДИНАМІЧНОГО СЕРЕДОВИЩА ГЕОМЕТРИЧНОГО МОДЕЛЮВАННЯ ДЛЯ МОБІЛЬНИХ ТЕЛЕФОНІВ

Метою бакалаврської роботи є створення динамічного середовища геометричного моделювання для мобільних телефонів.

Об’єктом дослідження є комп’ютерні технології створення динамічного середовища.

Предметом дослідження є комп’ютерні технології геометричного моделювання криволінійних поверхонь для мобільних пристроїв.

Для досягнення цієї мети, потрібно виконано наступні завдання:

— проаналізувати існуючі методи та засоби для створення динамічного середовища геометричного моделювання;

— обрати методи та засоби для створення динамічного середовища геометричного моделювання;

— розробити структуру, діаграму класів, та алгоритм для керування поверхнею;

— розробити інтерфейс програмного забезпечення.

Вхідною інформацією вважаються координати опорних вершин поверхні Безьє.

Вихідною інформацією вважається змодельована поверхня Безьє.

Програмний продукт розроблено мовою програмування C# в середовищі Microsoft Visual Studio за допомогою спеціального ігрового рушія Unity3D. Для підключення основних можливостей Unity3D, було застосовна бібліотеку UnityEngine.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА ПРОГРАМНИХ ЗАБЕЗПЕЧЕНЬ ДЛЯ СТВОРЕННЯ ДИНАМІЧНОГО СЕРЕДОВИЩА ГЕОМЕТРИЧНОГО МОДЕЛЮВАННЯ

Порівнявши існуючі системи динамічного моделювання поверхні Безьє на основі ізотропних характеристик, можна прийти до висновку, що кожна з них є унікальною. Обирають такі системи, в залежності від специфікації галузі. Порівняємо розроблений програмний продукт, з системами доволі поширеними, а саме : “AutoCAD” та “Unigraphics”. Розглядатимемо обрані програмні засоби в таблиці 2.1.

	AutoCAD	Unigraphics	Розроблена система
Моделювання кривої Безьє другого порядку	+	+	+
Можливість динамічного змінення поверхні	-	+	+
Можливість побудови мінімальної поверхні	-	+	+
Можливість використовувати програмне забезпечення на мобільному телефоні.	+	-	+

Таблиця 2.1— Порівняння існуючих продуктів

Отже, виходячи з порівняння, можна зробити висновок, що розроблений програмний продукт, конкурентно спроможний.

2.1 Програмне забезпечення “AutoCAD”

Програмне забезпечення “AutoCAD” було створене фірмою Autodesk. Перші версії цього продукту орієнтувались на двовимірне креслення і випуск конструкторської документації, а з часом він перетворився на конкуренто спроможне середовище тривимірного моделювання. Ядро цієї системи, було написано на мові C++, що дає можливість фірмі Autodesk, підключати безліч прикладних програм, написаних на цій мої.

Даний програмний продукт реалізує 2D- і 3D-технології проектування, практично будь-якої сфери діяльності (рисунок 2.1).

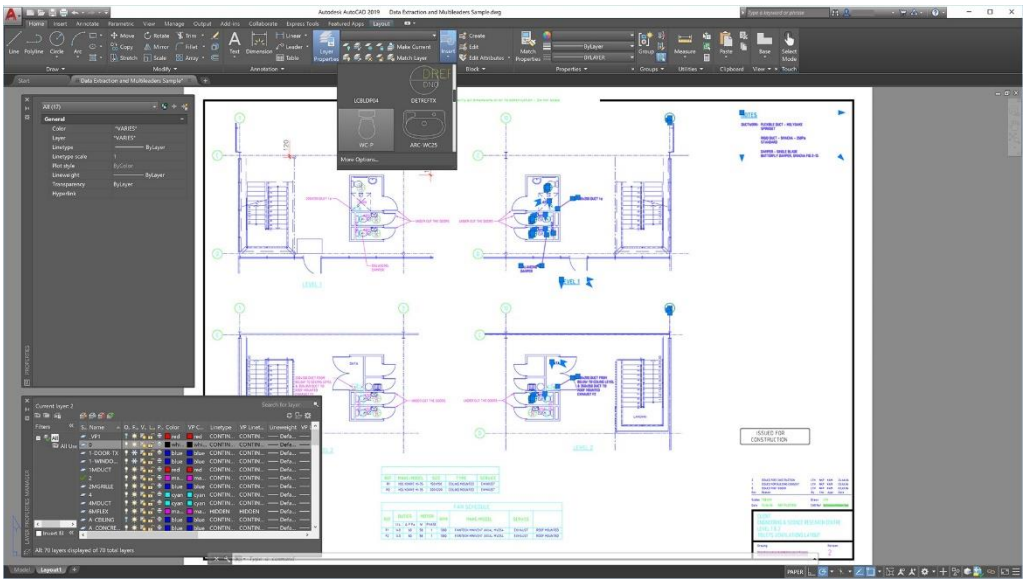

Рисунок 2.1 — вікно системи “AutoCAD”

Цікавим інтегрованим пакетом AutoCAD є Mechanical Desktop(AMD), тому що він включає можливість програмного продукту від компанії Autodesk – AutoSurf.

AutoSurf використовується для моделювання складних тривимірних поверхонь з використанням NURBS-геометрії. Для створення тривимірних моделей AutoCAD Designer пропонує виконати наступні етапи: спочатку задається плоский ескіз деталі, а потім йому додається третій вимір. При конструюванні складальної одиниці користувачеві потрібно задати параметричні зв'язки між об'єктами, обмежуючи число ступеней свободи проектованої механічної системи.

2.2 Програмне забезпечення “Unigraphics”

Система Unigraphics фірми EDS використовується в аерокосмічній і автомобільній промисловості, а також в машинобудуванні. Основна особливість цього програмного забезпечення — це наявність засобів гібридного тривимірного моделювання. За це відповідає модуль Solid Modeling, який дозволяє моделювати поверхні. Також цікавість викликає модуль Freeform Modeling який відповідає за тривимірне моделювання складних “скульптурних” поверхонь.

На рисунку 2.2 зображено інтерфейс цього програмного забезпечення.

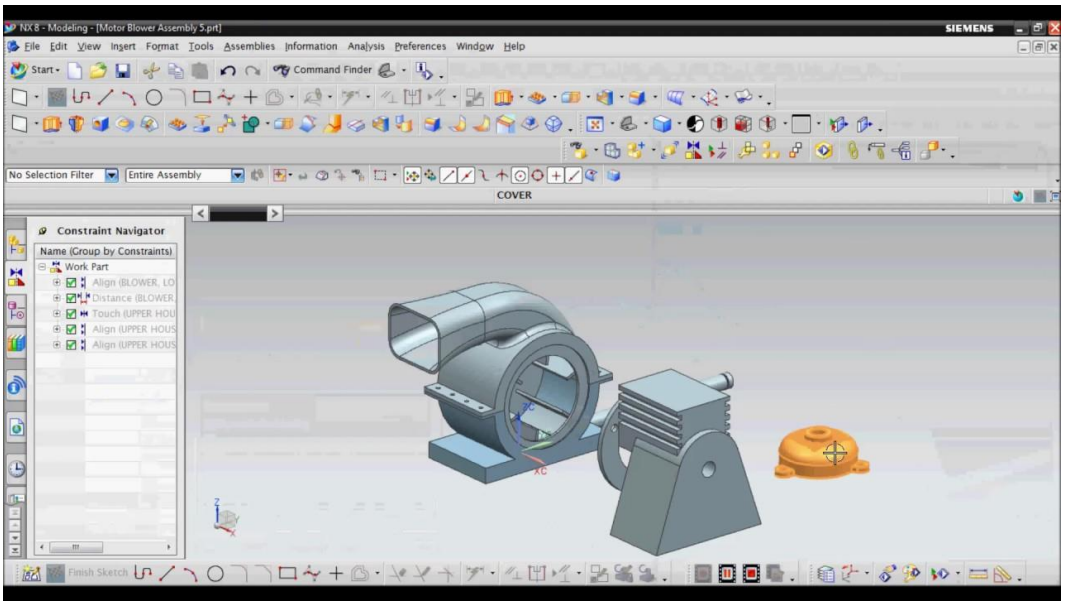

Рисунок 2.2 — вікно системи Unigraphics

Система вирізняється з-поміж інших завдяки додатку Mechatronics Concept Design, який на ранніх стадіях проектування дозволяє виконати початкові фізичні перевірки і симулювати працездатність конструкції.

3 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ГЕОМЕТРИЧНОГО МОДЕЛЮВАННЯ

В даному розділі описані необхідна інформація для створення програмного забезпечення геометричного моделювання на мобільному пристрої. Об’єктом моделювання було обрано поверхню Безьє, яка базується на кривих Безьє другого порядку.

3.1 Технології геометричного моделювання

Можна виділити два види комп’ютерного геометричного моделювання: двовимірне(плоске) та тривимірне (просторове).

Для двовимірної характерне автоматизована побудова та оперування плоскими геометричними моделями. Двовимірне моделювання використовується передусім для створення графічних конструкторських документів.

Основні методи побудови таких моделей є методи нарисної геометрії та методи креслення графічних примітивів (відрізків, прямих, дуг, кіл). Також слід зазначити, що такий процес моделювання відбувається одночасно з розробленням конструкторської документації, що і відрізняє його від тривимірного моделювання, яке передує цьому процесу.

Тривимірне геометричне моделювання будує об’ємні моделі об’єктів у віртуальному тривимірному просторі.

Технології тривимірного геометричного моделювання дозволяють розширити сферу застосування геометричних моделей і дозволяють ефективно їх використовувати в функціональному та технологічному проектувані.

На сьогоднішній день існує три напрями тривимірного моделювання: каркасне, поверхневе та твердотільне. При каркасному моделюванні геометрія моделі описується контурами та ребрами, що лежать на поверхнях деталі. Поверхнева модель відображає форму деталі за допомогою поверхонь, які обмежують її модель. Головна відмінність твердотільного моделювання від поверхневого, це те, що в явній

формі містяться дані щодо приналежності елементів внутрішньому або зовнішньому по відношенню до деталі простору. Іншими словами, вона не є пустою всередині.

Завдяки цій технології комп'ютерна модель може володіти деякими властивостями фізичних моделей і може використовуватися для отримання різних механічних та технологічних розрахунків.[1]

3.2 Методи моделювання кривих

У цьому підрозділі описано базові принципи комп'ютерного моделювання плоских кривих.

3.2.1 Базові поняття для створення кривих

Застосування кривих ліній, охоплює різні сфери будівництва та майже всі галузі техніки. Без їхнього застосування, не можливо уявити розв'язання різноманітних наукових та інженерних задач. Найбільш часто криві застосовуються у геометричному моделюванні різних технічних об'єктів.

Кривими лініями можна описати обводи літаків, автомобілів, лопаток турбін і не тільки. Вони допомагають розв'язувати певні наукові або інженерні завдання, при цьому криві подаються, як графічно, так і рівнянням у системі координат. Тому, наведемо існуючі способи створення кривих:

— крива визначена, як геометричне місце точок (наприклад, Равлик Паскаля);

— крива визначена, як траєкторія руху точки (наприклад спіраль Архімеда, точка якої бере участь у двох рухах — пряма та коло);

— крива визначена як лінія в результаті деякого геометричного перетворення вже відомої кривої;

— крива визначена як лінія перетину певної поверхні площиною, положення якої визначене (наприклад, криві 2-го порядку визначаються, як перетин кругового конусу).

Не дивлячись на те, що вже існує достатньо різноманітних методів моделювання кривих, продовжують існувати задачі, які продовжують вимагати нових способів формування кривих. Поява нових методів моделювання кривих є основним рушієм

для розширення можливостей комп’ютерної техніки для відображення графічних даних. Комп’ютерна графіка широко використовується не лише в проектуванні техніки, але й в кіно. На сьогоднішній день, спецефекти в кіно досягли таких висот, про які раніше навіть ніхто не уявляв.[2]

3.2.2 Основні поняття про криву

Крива — це неперервна послідовність точок в просторі, що рухаються. В інженерній графіці криві характеризують за іх проекціями. Крива називається просторовою, якщо точки кривої не належать одній площині.

Порядок кривої лінії визначається степенем рівняння. З геометричної точки зору порядок плоскої алгебраїчної кривої характеризується числом точок (дійсних та уявних) перетину її з прямою лінією. Порядок просторової кривої характеризується кількістю точок перетину цієї лінії з площиною.[3]

Типи кривих:

— гладкі;

— з точкою зламу;

— з точкою перегину;

— з подвійними точками.

Гладкі криві — це криві в яких радіус кривизни плавно з одного значення в інше.

Криві з точкою зламу — це криві, які мають таку точку в якій не можна провести дотичну, а радіус кривизни не має ніякого значення.

Криві з точкою перегину — це криві, які мають таку точку в яких радіус кривизни має нескінченне значення.

Криві з подвійними точками — це криві, які мають точки повертання кривої.

3.3 Моделювання поверхні Безьє

3.3.1 Крива Безьє

Сплайни Безьє являють собою інструмент для проектування просторових конструкцій. Вони використовуються в системах автоматизованого проектування різних механізмів. Криві Безьє використовуються в комп’ютерній графіці та

картографії, для побудови маршруту з заданими початковими і кінцевими пунктами, і який який проходить ще через декілька пунктів розташованих поруч.

Окрім плоских кривих Безьє, використовують також і просторові поверхні Безьє. Поверхня приваблює тим, що згладжує нерівності просторової фігури. Прикладом може слугувати вся поверхня кузова автомобіля. Для досягнення потрібного візуального враження, достатньо змінити координати деяких точок, що досягається швидко і зручно.[4]

Параметричне рівняння кривої Безьє у загальному вигляді:

$$B(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i \ , \ t \in [0,1], \tag{3.1}$$

де P_i — опорні вершини,

$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$ — поліноми Берштейна

3.3.2 Властивості поверхні Безьє

Крива Без'є має такі властивості:

- якщо контрольні точки кривої знаходяться на одній прямій, то крива буде визначена як пряма лінія;
- крива знаходиться мжу лініями, що поєднують контрольні точки;
- не існує пробілів між початковою і кінцевою точками кривої;
- зміна напрямку траєкторії не впливає на форму кривої;
- байдужа до афінних перетворень;
- будь-яка зміна координат опорних точок, веде до зміни форми всієї кривої;
- будь-який сегмент кривої Безьє можна виразити через іншу криву Безьє;
- степінь кривої завжди на одиницю менший від кількості контрольних точок[5].

Виділяють три види кривих Безьє, такі як лінійні криві Безьє, квадратичні, кубічні. В даній роботі розглянуто квадратичні криві Безьє, оскільки цей тип дозволяє створити програмний засіб, який задовольняє поставленій меті, і він потребує меньшої кількості обчислень ніж кубічні, що має найбільший пріоритет, так як

програмне забезпечення буде використовуватись на мобільному пристрої.

Загальне рівняння кривої Без'є другого порядку має наступний вигляд

$$r = p_0(1-u)^2 + 2p_1(1-u)\ u + p_2u^2,\ u \in [0,1], \qquad (3.2)$$

де p_0, p_1, p_2 — координати опорних вершин.

Три опорних точки, заданих в просторі визначають форму кривої(рисунок 3.1).

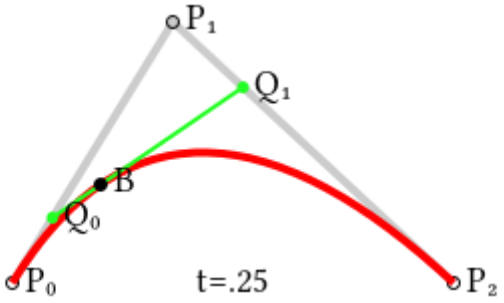

Рисунок 3.1 — Крива Безьє другого порядку

Крива починається в точці P_0 поступово направляється до P_1 і закінчується в точці P_2. Крива не проходить через ці точки, вони лише вказують напрям руху.

Для побудови кривої Безьє другого порядку потрібно визначити координати трьох опорних вершин. Дві точки p_0 і p_2 виступають, як початок і кінець кривої відповідно, а інша p_1, відіграє роль направляючої. Для кращого сприйняття, в програмах, контрольні точки з'єднують лініями, утворюючи таким чином характеристичний трикутник (рисунок 3.2).

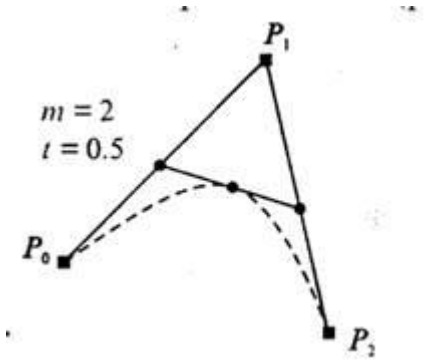

Рисунок 3.2 — Крива Безьє другого порядку і допоміжний характеристичний трикутник

Отже, за допомогою переміщення цих трьох точок можна отримати велику кількість різних форм кривих Безьє, які можуть бути частиною більш складного контуру.

В будь-якому сегменті можна назначити нові контрольні точки, які також впливатимуть на форму кривої. Нові контрольні точки в межах сегмента кривої не заперечує факт, що окремі криві поєднуються в ланцюг. Тому що крива Безьє знаходиться в середині вже існуючого контуру.

Будь-який векторний контур або векторна форма задаються за допомогою векторних сегментів, які є ідентичними окремій простій кривій Безьє.[6]

3.4 Побудова поверхні Безьє

Поверхні Безьє набули великої популярності в комп'ютерній графіці, оскільки вони набагато компактніші, простіші в маніпуляції по відношенню до трикутної сітки і вони мають гарні властивості безперервності(рисунок 3.3).

Поверхня Безьє порядку n, m буде задаватися (n + 1)(m + 1) контрольними точками. Вона представляє гладку безперервну поверхню і находиться у тому ж просторі що і контрольні точки. Якщо контрольні точки в тривимірному просторі, то і поверхня буде в тривимірному просторі. Поверхня Безьє в двовимірному просторі, може бути визначена як параметрична поверхня, і обчислюється по наступній формулі:

$$p(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u)\, B_j^m\, Pij \tag{3.3}$$

де $u, v \in [0,1]$, В – многочлени Бернштейна.

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} = \frac{n!}{i!(n-i)!} u! \,(1-u)^{n-i} \tag{3.4}$$

Властивості поверхонь Безьє:

— поверхня Безьє реагує на будь-які зміни її опорних точок при всіх лінійних перетвореннях і зрушеннях;

— поверхня Безьє буде повністю лежати в опуклої області своїх опорних точок в будь-якій декартовій системі;

— як правило поверхня Безьє не проходить через всі свої контрольні точки.

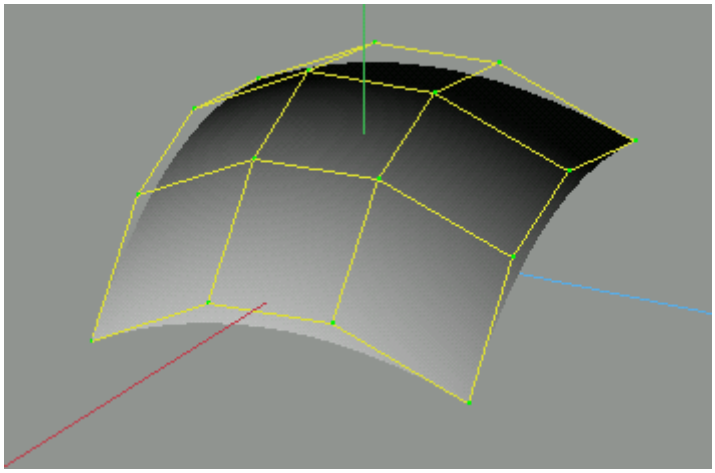

Рисунок 3.3 — Змодельована поверхня Безьє

Для моделювання поверхонь із визначеними диференціальними властивостями було розглянуто методи моделювання плоских ізотропних сіток. Для побудови просторових сіток необхідно розширити можливості щодо моделювання кривих, а саме розглянути моделювання просторових кривих

3.5 Двомірні геометричні перетворення

Будь-яке програмне забезпечення для графічного моделювання, має функції для виконання геометричних перетворень, таких як: трансляція, обертання і масштабування. Для того що б зрозуміти загальні концепції, пов’язані з геометричним перетворенням, спочатку розглянемо їх у двомірному просторі, після чого перейдемо до тривимірного.

3.5.1 Трансляція

Трансляція — це процес переміщення точки, який виконується шляхом додавання зміщення до її координат. Тобто точка просто переміщується по прямій в нове місцезнаходження. Подібним чином транслітерація діє і на об’єкт, який визначається множиною точок (наприклад, трикутник), вона переміщує всі точки об’єкту на однакову відстань вздовж паралельних прямих. Тим самим відображаючи об’єкт в іншому місці.

При трансляції двомірної точки до початкових координат (x, y) додається приріст u_x і u_y в результаті чого ми отримуємо нову точку, як показано на рисунку 3.4.

$$x' = x + u_x, \quad y' = y + u_y. \tag{3.5}$$

Пара приросту (u_x, u_y) називається вектором здвигу.

Вираз трансляції (3.5) можна записати у вигляді матричного рівняння, в якому будуть виступати приведені нижче векторами-стовцями, які виражають координати точок і вектори трансляції.

$$B = \begin{bmatrix} x \\ y \end{bmatrix}, \quad B' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad U = \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{3.6}$$

Такі позначення, дозволяють записати рівняння двомірної трансляції в матричній формі наступним чином:

$$B = B' + U \tag{3.7}$$

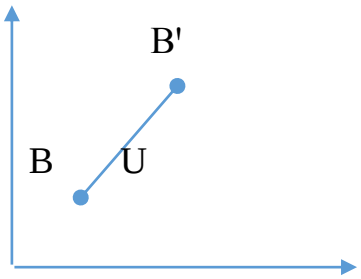

Рисунок 3.4 — Трансляція точки з B в точку B' з використанням вектору трансляції U

3.5.2 Обертання об’єкту

Для того щоб обернути об’єкт потрібно задати вісь і кут обертання. Після чого всі точки об’єкту переміщуються на нове місце шляхом обертання точок на заданий кут навколо осі обертання.

Обертання об’єкту виконується шляхом переміщення його по круговій траєкторії на площині xy. В результаті чого об’єкт обертається відносно осі перпендикулярній площині xy(паралельній координатній осі z). Щоб спростити пояснення, визначимо рівняння обертання точки P, коли центром обертання, являється початок координат. Розглянемо наступне рівняння:

$$x' = rcos(\varphi + \theta) = rcos\varphi cos\theta - rsin\varphi sin\theta;$$
$$y' = rsin(\varphi + \theta) = rcos\varphi sin\theta - rsin\varphi cos\theta. \quad (3.8)$$

Де r – постійна відстань точки від початку координат, кут $\varphi -$ початкове кутове положення точки відносно горизонтальної лінії, $\theta -$ кут обертання.

Вихідні полярні координати точки відповідно дорівнюють:

$$x = rcos\varphi, y = rsin\theta \quad (3.9)$$

Підставивши вираз (3.9) в (3.8), отримуємо рівняння перетворення для обертання точки з координатами (x, y) на кут θ навколо початку координат:

$$x' = xcos\theta - ysin\theta;$$
$$y' = xcos\theta - ycos\theta \quad (3.10)$$

Використовуючи представлення через вектори-стовці (3.6), рівняння повороту можна записати у матричній формі:

$$B' = R \cdot B, \quad (3.11)$$

де матриця обертання має наступний вигляд:

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}. \quad (3.12)$$

3.6 Тривимірні геометричні перетворення

Методи тривимірних перетворень насправді являються розвинутими методами двовимірних в які добавлено додаткову інформацію, яка відноситься до третьої координати z. Трансляція об’єкта виконується тепер за допомогою тривимірного вектору трансляції, який і визначає, на скільки потрібно перемістити об’єкт по кожній з трьох координат. Але розширення методів двомірних обертань в тривимірні не так прямолінійно, що буде продемонстровано далі.

3.6.1 Тривимірна трансляція

Точка B з координатами (x, y, z) тривимірного простору транслюється в точку B' з координатами (x', y', z') в наслідок додавання відстані трансляції $u_x u_y u_z$ до декартових координат точки B:

$$x' = x + u_x, \quad y' = y + u_y, \quad z' = z + u_z. \tag{3.13}$$

Тривимірну трансляцію точки можна побачити на рисунку 3.5.

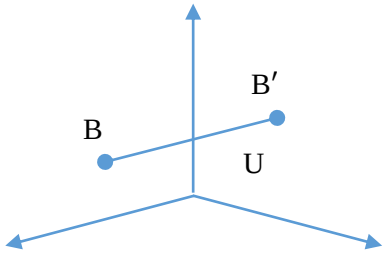

Рисунок 3.5 — Дія на точку B вектору трансляції U.

Тепер виразимо дану трансляцію в матричній формі, аналогічно як і для двомірної . Точки B і B' являються в однорідних координатах чотириелементними векторами-стовпчиками, а оператор трансляції T — це матриця 4 на 4:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 100u_x \\ 010u_y \\ 001u_z \\ 000\ 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.14}$$

або

$$B' = T \cdot B \tag{3.15}$$

Для того щоб транслювати об’єкт в трьох вимірах потрібно змінити всі координати об’єкта, після чого відновити об’єкт в новому місці. Якщо взяти об’єкт який являється набором багатокутних поверхонь, тоді трансляція виконується над усіма його вершинами і вкінці будуються багатокутні грані.

3.6.2 Тривимірне обертання

Об’єкт можна обертати навколо будь-якої осі в просторі, але найпростіше буде обробляти обертання навколо осей, паралельних до декартових. Окрім того, ще можна використовувати комбіноване обертання координатних осей, щоб задати поворот навколо будь якої іншої лінії простору.

Рівняння двомірного повороту навколо осі z легко можна подати і для тривимірного простору:

$$\begin{aligned} x' &= xcos\theta - ysin\theta, \\ y' &= xsin\theta + ycos\theta, \\ z' &= z. \end{aligned} \tag{3.16}$$

Параметр θ задає кут обертання навколо осі z, а значення координати z при цьому не змінюється. Рівняння тривимірного обертання навколо осі z можна записати через однорідні координати, наступним чином:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.17}$$

або можна записати наступним чином:

$$B' = R_z(\theta) \cdot B. \tag{3.18}$$

Рівняння обертання навколо двох інших координатних осей можна отримати за допомогою циклічної перестановки параметрів x, y і z в рівняннях (3.16).

4 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При розробці програмного продукту важливим чинником є правильний вибір технологій та засобів програмної реалізації. Основним середовищем розробки було Microsoft Visual Studio 2015, а також Unity3D. Для створення інтерфейсу використовувалися засоби Unity3D.

Для розробки алгоритмів використовувалась мова C#.

За контроль версії відповідав Git і Unity Asset Server.

Для використання всіх засобів Unity3D, була використана бібліотека UnityEngine.

4.1 Середовище розробки Visual Studio 2017

Середовище розробки Visual Studio дозволяє швидко і ефективно писати код, не втрачаючи з уваги контекст поточного файлу (рисунок 4.1). Можна легко заглибитися в подробиці, такі як структура виклику, пов'язані функції, повернення і стан тестування. Також доступні рефакторинг коду, знаходження і усунення помилок в коді [7].

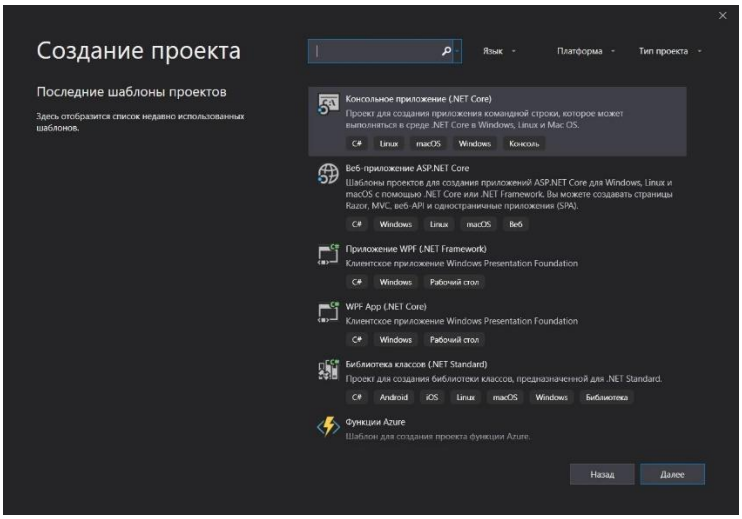

Рисунок 4.1 — Початкове вікно середовища розробки Visual Studio

Сьогодні сімейство інструментів Visual Studio 2017 містить IDE, сервіс для організації спільної роботи - Visual Studio Team Services, комплексне рішення для реалізації повноцінного циклу розробки мобільних додатків - Visual Studio Mobile Center, багатоплатформовий редактор коду Visual Studio Code (доступний для Mac, Linux і Windows), а також пробна-версія Visual Studio for Mac.

З кожною версією інструментів Microsoft намагається врахувати побажання розробників і зробити їх зручніше для створення додатків практично для будь-якої платформи. Результатом є величезний інтерес і більше 21 млн установок інструменту на сьогоднішній день.

По-перше, вже зараз абсолютно будь-який розробник може завантажити собі повноцінну версію Visual Studio 2017 і отримати 60-денну безкоштовну передплату для доступу до Xamarin University - навчає сервісу про створення кроссплатформенних мобільних додатків на C #.

По-друге, творці продовжують піклуватися про підвищення продуктивності розробників, створюючи всі умови, щоб сконцентруватися тільки на написанні коду. Наприклад, поліпшення в уже полюбилися можливості навігації за кодом, рефакторінга, виправлення і налагодженні для всіх підтримуваних мов. Додатково, нова версія дозволяє збільшити швидкість командної розробки з новими real-time функцій модульного тестування і перевірки залежностей.

Третя важлива зміна торкнулася процесу установки інструменту. Новітній, полегшений модульний підхід дозволяє вам встановити тільки ті компоненти середовища, які необхідні і прискорює установку інструменту від початку і до кінця. До того ж, тепер у розробників пропала необхідність створювати проекти і рішення, щоб налагодити будь-який необхідний фрагмент коду.

Останні презентації Visual Studio не обійшлися без демонстрації поліпшень інтеграції з сервісами хмарної платформи Azure. Розробки Microsoft в цьому напрямку дозволяють полегшити створення, налагодження, розміщення і публікацію ваших додатків в хмарі Azure прямо з IDE, надаючи до того ж вбудовані інструменти для роботи цими додатками, а також з Docker-контейнерами, .NET Core додатками і так далі.

Інша важлива зміна на стороні мобільного розробки. Розробники отримали поліпшені інструменти налагодження і профілювання, інструменти генерації модульних тестів. І якщо ви плануєте створювати кроссплатформне додаток, то зараз настав той самий час, коли варто подивитися в бік Visual Studio 2017 і Xamarin, або використовувати альтернативний підхід з Apache Cordova, а можливо і Visual C ++, але вже для створення кроссплтаформенних бібліотек в рамках того ж інструменту - Visual Studio 2017.[7]

4.2 Ігровий рушій Unity3D

Unity — це інструмент, який підтримує різні платформи, а також дає змогу розробляти тривимірні додатки та ігри (рисунок 4.2). Системи, які підтримують роботу застосунків створених в Unity наступні: Android, iOS, Windows, а також на гральні консолі PlayStation 3 і XBox 360.

Дозволяє цей ігровий рушій також створювати інтернет-додатки за допомогою модуля для браузера Unity. Також в Unity існує підтримка DirectX та OpenGL. [11]

Unity підтримує лише дві мови: C # та JavaScript. Будь який проект в Unity починається з сцени. Сцена – це окремі файли, які містять свої ігрові світи, мають свій набір об’єктів та налаштувань. Об'єкти, в свою чергу містять компоненти, з якими і взаємодіють скрипти. Кожен об’єкт має свою назву, може бути створений тег (мітка) і шар, на якому він повинен відображатися. [8]

У будь-якого предмета на сцені обов'язково присутній компонент Transform — який відповідає за розташування, поворот і розмір об’єкта у просторі по всіх трьох осях. У об'єктів з видимою геометрією за умовчанням існує компонент Mesh Renderer і MeshFilter, що робить модель видимою.

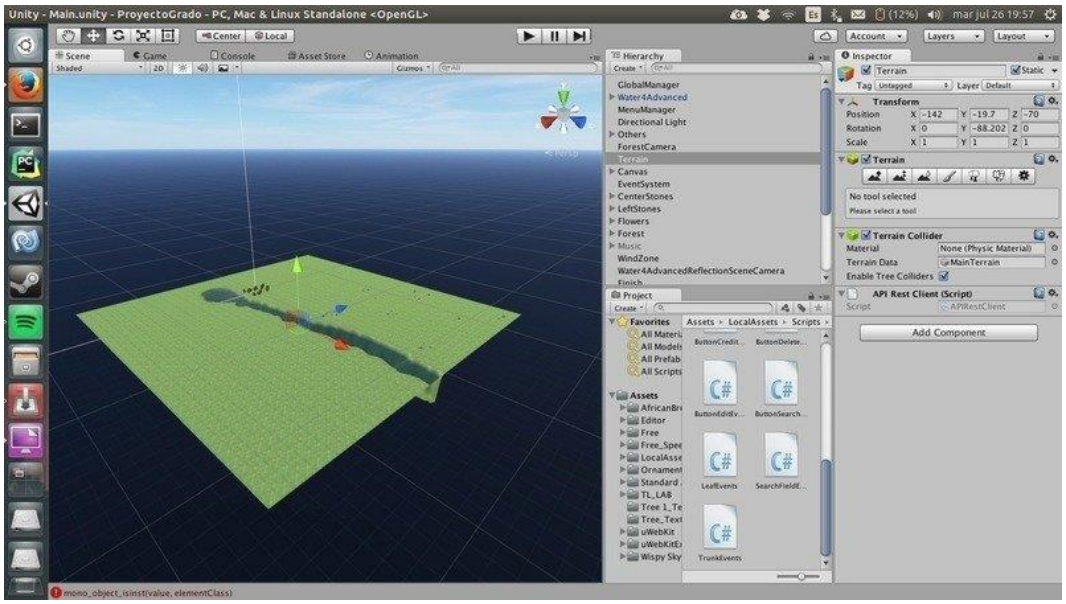

Рисунок 4.2 — середовище розробки Unity3D

4.3 Система контролю версій

Система контролю версій дозволяє зберігати файли та завантажувати їх за потребою. Деякі системи зберігають версії файлів, а деякі, роблять знімок змін, які відбувалися і змінюють початковий файл. Ці системи, дозволяють не хвилюватись про наслідок зміни, якоїсь частини коду, тому що дозволяють повертатись до попередньої версії, без жодних наслідків. На сьогоднішній день існує дві головних системи для керування репозиторіями: Git та Mercurial.

Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, які в свою чергу базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. Найбільш використовуваними представниками даної системи можна назвати GitHub та GitLab.

В даному проекті було обрано платформу GitHub, що є найпопулярнішою системою керування репозиторіями програмного коду. Репозиторій з програмним кодом системи має публічний доступ.

В Unity також існує своя система контролю версій Unity Asset Server для ігрових об'єктів та скриптів. Що дозволяє легко керувати багатогігабайтними проектами з тисячами мегабайтних файлів.[12]

Будь які зміни ресурсів одразу видні в редакторі Unity. Будь які зміни файлу, негайно оновлюють їх статус. Переміщення, або перейменування ресурсу не повинно створювати будь-яких перешкод для безперервного робочого процесу.

4.4 Бібліотека класів UnityEngine

UnityEngine – простір імен, який надає класи, що дозволяють взаємодіяти з Unit3D процесами, журналами подій і лічильниками продуктивності.

MonoBehaviour – це базовий клас, від якого успадковуються всі скрипти.

При використанні Javascript кожен скрипт автоматично успадковується від MonoBehaviour. Коли використовується C# або Boo Вам необхідно явно успадковуватися від MonoBehaviour.

Існує два основних методи, які використовуються при роботі з Unity3D, це Update() і Start().

Метод Start() викликається у фреймі, як будь-який з методів почне викликатися в перший раз.

Як і функція Awake, Start викликається рівно один раз за час існування скрипта. Однак Awake викликається при ініціалізації об'єкта сценарію, незалежно від того, чи включений сценарій. Запуск не може бути викликаний в тому ж кадрі, що і Awake, якщо сценарій не включений під час ініціалізації.[13]

Функція Awake викликається на всіх об'єктах в сцені перед викликом функції Start будь-якого об'єкта. Цей факт корисний у випадках, коли код ініціалізації об'єкта A повинен покладатися на вже ініціалізіруемий об'єкт B; Ініціалізація B повинна бути зроблена в Awake, а A - в Start.

Метод Update() вызывается каждый кадр, если MonoBehaviour включен.

Що б отримати час, який пройшов з останього виклику Update, можна використати Time.deltaTime. Ця функція викливається тільки, якщо Behaviour

включений. Можна переоприділити цю функцію, що б добавити до компоненту функціональності.[14]

4.5 Архітектура програми

Система динамічного середовища геометричного моделювання, являється відкритою і може доповнюватись новими модулями. Архітектуру системи наведено на рисунку 4.3.

Користувач спілкується з екраном телефону, який виконує функції керівного модуля. Через екран виконується зв'язок з інтерфейсною, обчислюваною частинами та програмною візуалізацією.

В обчислюваний модуль на вході подається набір опорних точок, вони дістаються з збереженого файлу, або з файл по замовчуванню.

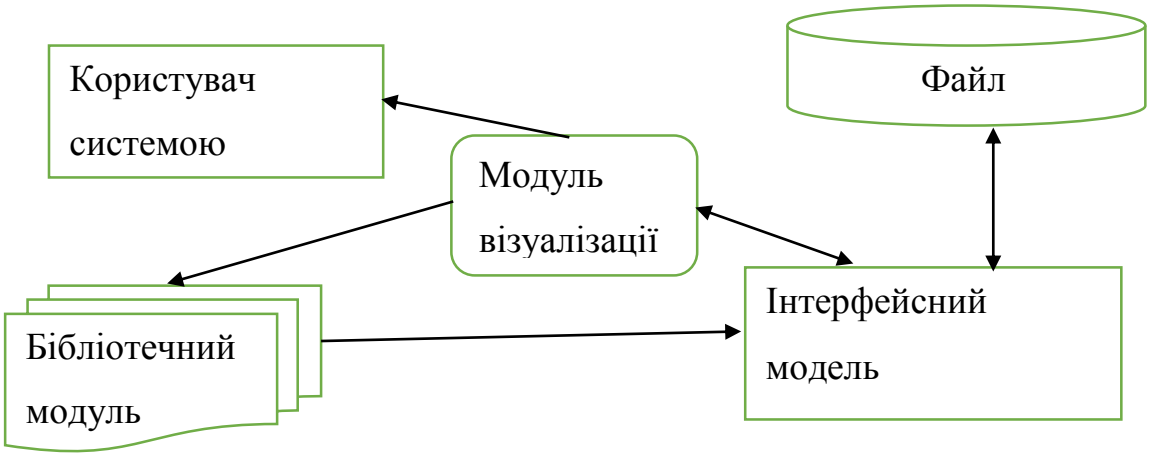

Рисунок 4.3 — Архітектура система.

4.6 Діаграма класів

Діаграма класів програмної системи динамічне середовище геометричного моделювання для мобільних телефонів на основі кривих Безьє другого порядку представлена на рисунку 4.4. Діаграма класів відтворює структуру розробленого програмного забезпечення, демонструє основні ідеї об’єктно-орієнтованого програмування, ключові парадигми якого були втілені при реалізації системного продукту.

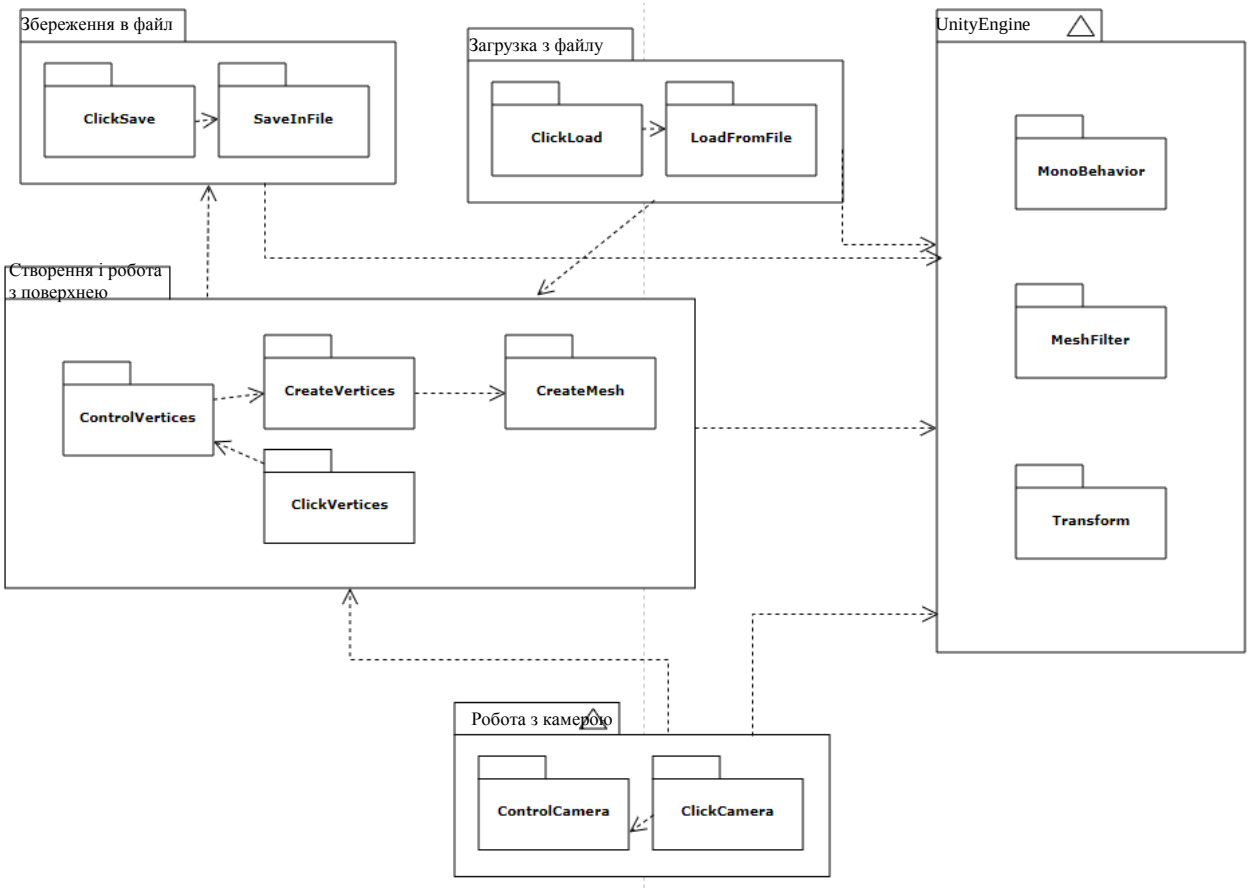

Рисунок 4.4 — Діаграма класів

4.7 Програмний модуль побудови поверхні Безьє і роботи з цією поверхнею

Побудова поверхні Безьє базується на множині кривих Безьє другого порядку зі значенням параметра u близько 0.1, які в свою чергу базуються на множині ліній зі значенням параметра v близько 0.1. Для цього було написано власний клас “CreateMesh”, який включає в себе клас “Point3D” і містить дев’ять полів типу “Point3D” для визначення дев’яти опорних вершин поверхні.

Для роботи з поверхнею, а саме динамічному зміненню її, було створено три додаткових класи, а саме: “CreateVertices”, “ControlVertices”, “ClickVertices”. Клас “CreateVertices” відповідає за створення опорних точок на сцені. Його робота заключається в отримані даних, про розміщення цих точок з класу “CreateMesh” і на основі об’єкту Point, який було створено засобами Unity3D, візуалізувати їх на сцені.

Клас “ControlVertices” відповідає за переміщення контрольних точок, в результаті якого, перебудовується і поверхня. За відслідковуванням натискання відповідає клас “ClickVertices”.[11]

4.8 Програмний модуль роботи з камерою

Забезпечивши створення модуля роботи з камерою, була надана можливість переміщення опорних точок у всіх площинах. В залежності від того, під яким кутом до поверхні знаходиться камера, обирається вісь по якій буде рухатись вершина. Клас “ClickCamera” відповідає за активування режиму керування камери і його припинення. За всі інші режими роботи з камерою, відповідає клас “ControlCamera”. Одним із його завдань являється відслідковування натискань на екран мобільного пристрою та переміщення камери в залежності від дій користувача.

4.9 Програмні модулі роботи з файловою системою

Програмний модуль дає можливість отримувати опорні точки з файлу і дозволяє зберігати нові опорні точки в інший файл. Даний програмний модуль ділиться на дві частини, які відповідають за збереження і завантаження даних. Для забезпечення роботи з збереженням даних було створено клас “SaveInFile”, який перетворює координати опорних точок в потрібний вигляд і записує їх в файл. Для роботи по завантаженню даних з файлу, розроблено клас “LoadFromFile”, даний клас парсить строки з файлу, і для кожної окремої строки, в якій знаходяться координати, створює новий об’єкт типу Point3D.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для забезпечення безвідмовної роботи програмної системи «Динамічне середовище геометричного моделювання для мобільних телефонів» потрібно дотримуватися основних вимог при інсталяції та рекомендацій щодо її використання.

5.1 Інсталяція та системні вимоги

Для роботи програмного продукту, користувач повинен мати мобільний пристрій який відповідає наступним вимогам:

— процесор MediaTek MT6737;
— оперативна пам’ять від 2Гб;
— вільний простір на жорсткому диску 1ГБ.

Вимоги до програмного забезпечення:

Android 5.1 / iOS 8.0.

Для інсталяції програмного продукту, потрібно знайти місце знаходження файлу програми на пристрої, це може бути папка Завантаження, далі потрібно натиснути на значок, який зображено на рисунку 5.1

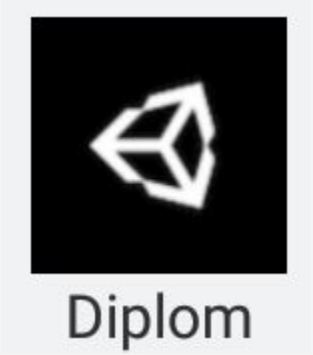

Рисунок 5.1 – Програма інсталяції

Після чого, операційна система запросить доступ на встановлення додатку з невідомого джерела (рисунок 5.2)

Рисунок 5.2 – Доступ на встановлення

Потрібно натиснути кнопку Настройки і навпроти пункту Невідомі джерела натиснути на дозвіл, як показано на рисунку 5.3

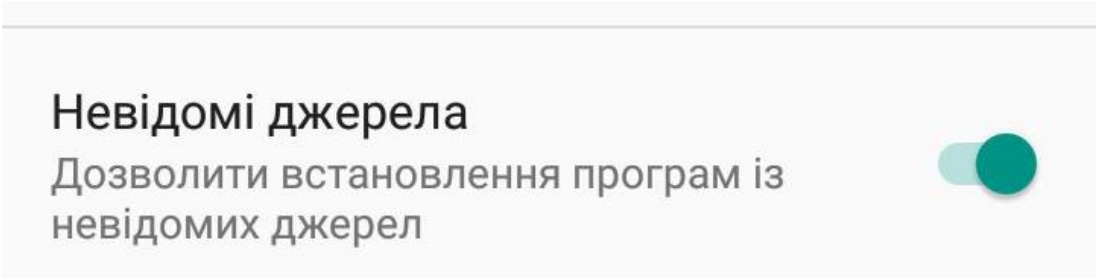

Рисунок 5.3 – Дозвіл на невідомі джерела

Після цих маніпуляцій, повернутись до встановлення додатку і просто натиснути кнопку Установити, яку зображено на рисунку 5.4

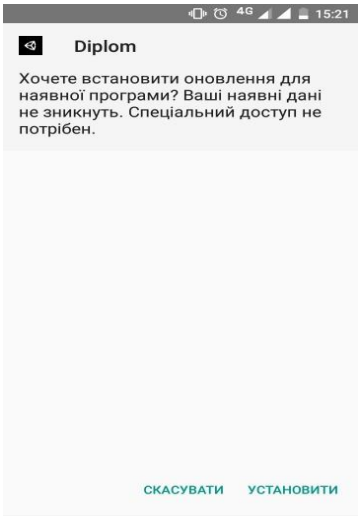

Рисунок 5.4 – Продовження встановлення

Наступним кроком, після встановлення додатку, потрібно натиснути кнопку відкрити і дозволити програмному продукту отримувати доступ до файлів збереження (рисунок 5.5).

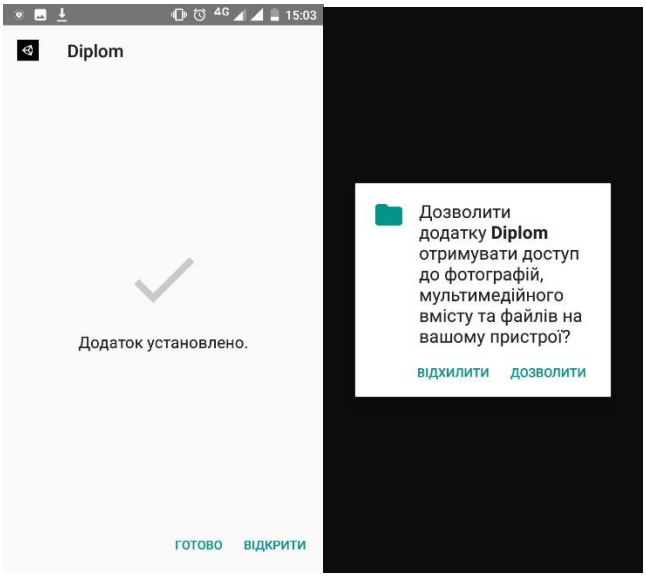

Рисунок 5.5 – Відкривання додатку.

При першому вході на екрані з’являється поверхня по замовчуванню(рисунок 5.6).

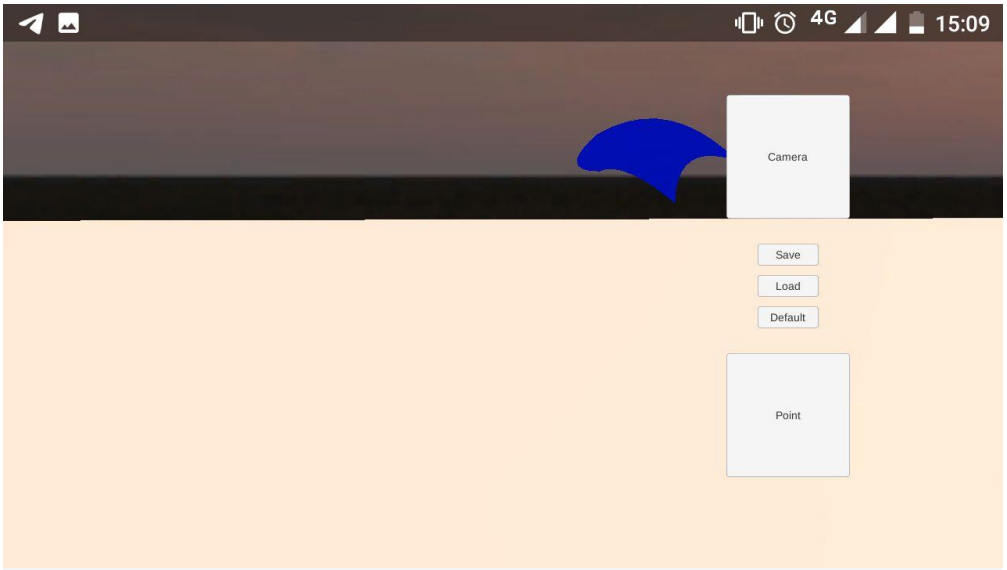

Рисунок 5.6 – Перший запуск програмного продукту.

Програмний продукт дозволяє імпортувати поверхні у форматі .txt, а також використовувати результати проміжного програмного забезпечення, наприклад, для відслідковування положення точок, шляхом виконання згенерованого ним скрипта.

Також, об’єктами деформації можуть бути змодельовані в даному середовищі об’єкти.

5.2 Сценарії роботи користувача з системою

Для початку роботи по деформації поверхні, можна використати поверхню по замовчуванню, для цього, достатньо натиснути кнопку Point, яка побудує контрольні точки, змінюючи положення яких, призводить до зміни поверхні (рисунок 5.7)

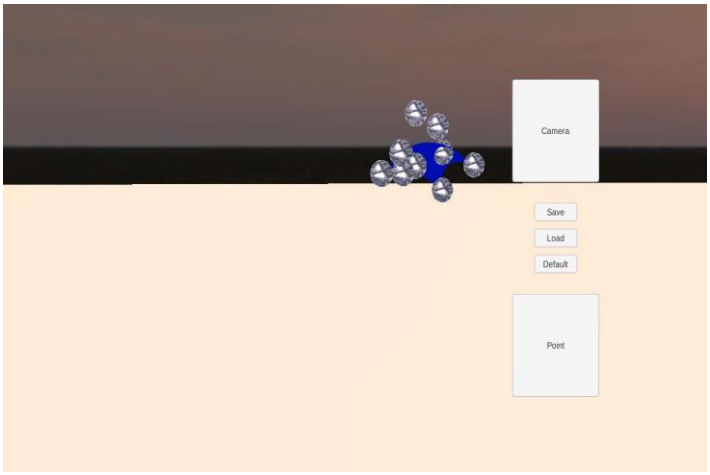

Рисунок 5.7 - Попередньо задана поверхня з контрольними точками

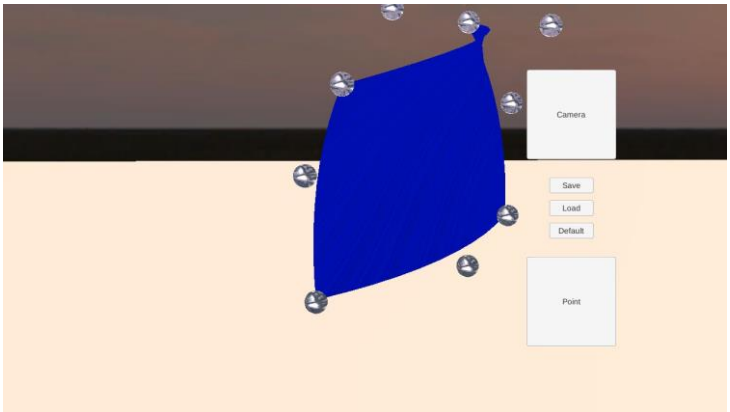

Рисунок 5.8 – Деформована поверхня

Після закінчення проектування поверхні її можна зберегти у файл з назвою Save.txt (нажавши кнопку Save), який зберігається на мобільному пристрої. Попередні дії, дозволять завантажувати деформовану поверхню при натисканні кнопки Load.

Також програмне забезпечення дозволяє керувати приближенням і віддаленням камери, а також дає можливість оглядати поверхню з будь-якої сторони, при цьому від положення камери залежить вісь, по якій ми можемо рухати контрольні точки (рисунки 5.9 – 5.11)

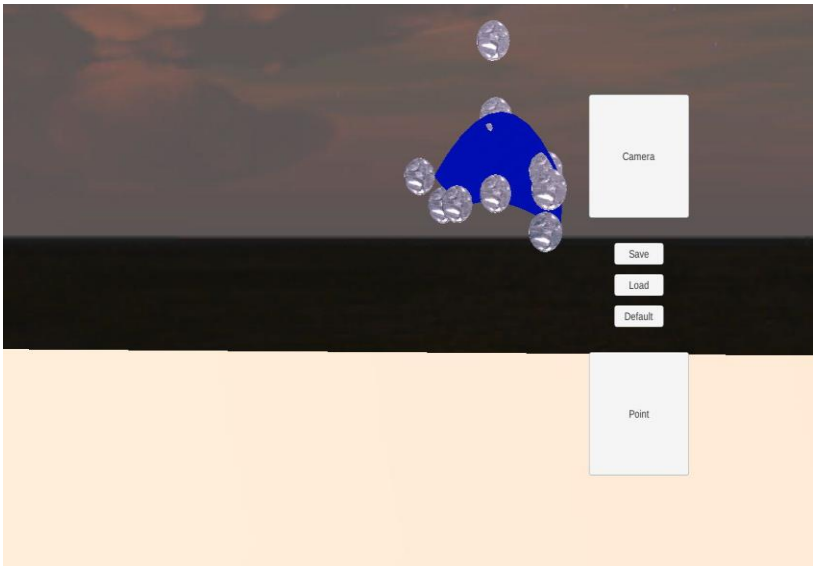

Рисунок 5.9 – Переміщення точки по осі XY

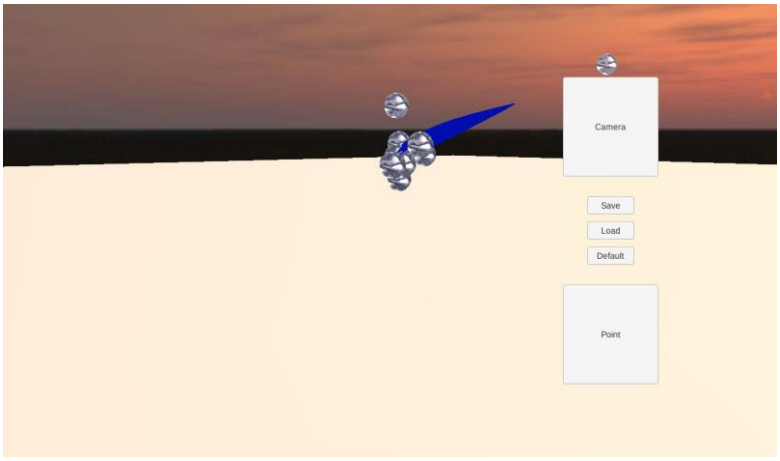

Рисунок 5.10 – Переміщення точки по осі ZY

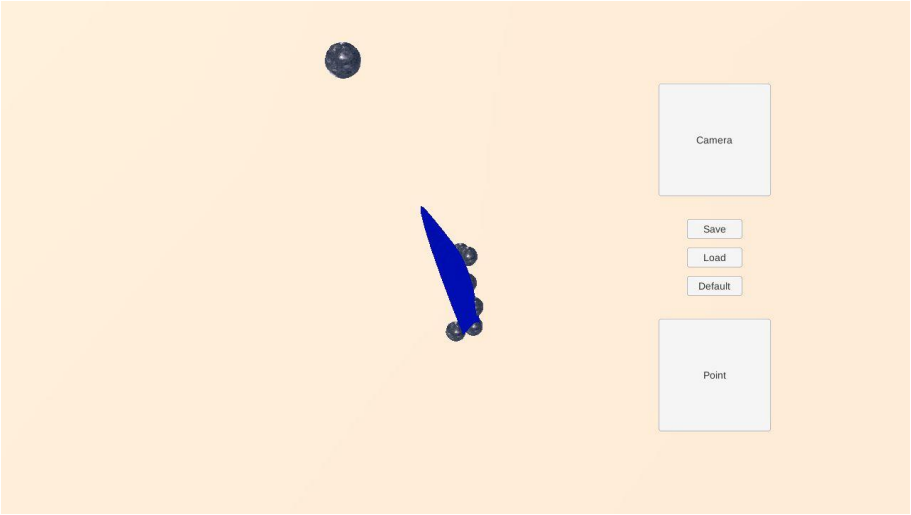

Рисунок 5.11 – Переміщення точки по осі XZ

5.3 Деінсталяція програмної системи

Якщо користувач закінчив роботу з програмним забезпеченням і бажає видалити його, йому необхідно виконати ланцюжок дій:

1. Знайти на телефоні знак шестигранника який зображений на рисунку 5.12.

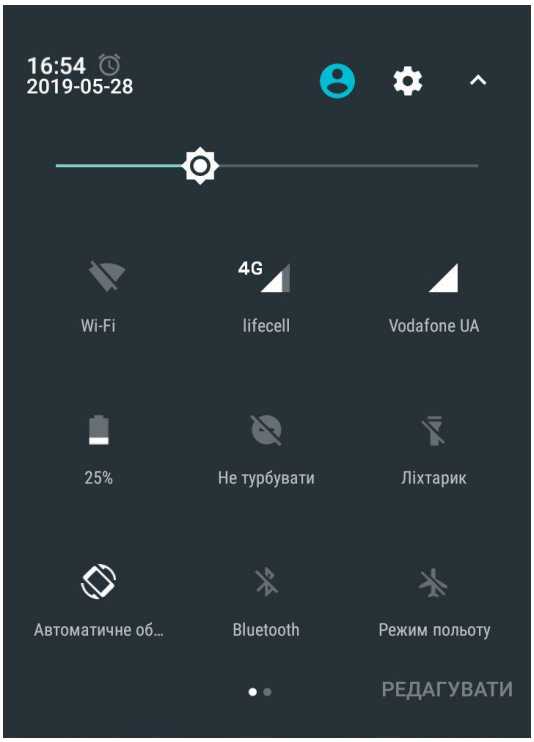

Рисунок 5.12 – Пошук меню налаштувань

2. У вікні налаштувань знайти пункт Додатки (рисунок 5.13).

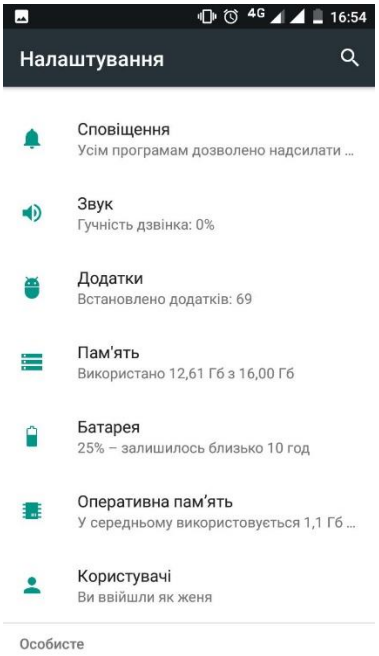

Рисунок 5.13 – Знаходження встановлених додатків

3. Знайти іконку програмного забезпечення, яке потрібно видалити (рисунок 5.14).

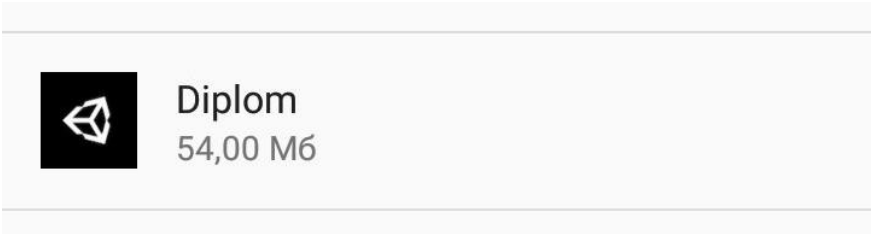

Рисунок 5.14 – Програмне забезпечення для видалення

4. Для видалення програми, натискаємо кнопку Видалити (рисунок 5.15)

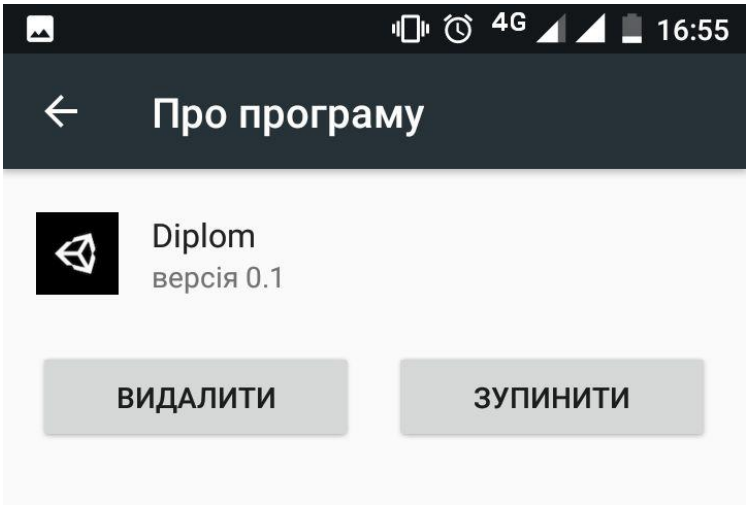

Рисунок 5.15 – Видалення програмного продукту

5. Погодитися на видалення програмного продукту (рисунок 5.16).

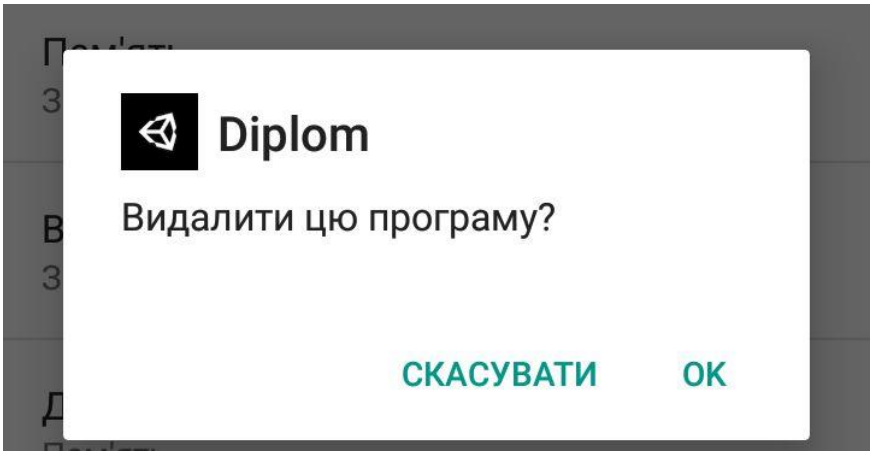

Рисунок 5.16 – Надання згоди видалення програмного модуля

ВИСНОВКИ

У ході виконання даної роботи було розроблено програмне забезпечення динамічного середовища геометричного моделювання для мобільних телефонів.

Було проаналізовано існуючі методи та засоби геометричного моделювання. На основі аналізу обрано метод та засіб для створення програмного забезпечення.

Під час написання програмного продукту, також було розроблено алгоритм для керування динамічною поверхнею Безьє.

Програмний продукт написано на мові C# та з використанням Unity3D.

Модуль є універсальним і може бути використаний для більш масштабного проекту, який спеціалізується на геометричному моделюванні.

Для демонстрації роботи розробленого модулю було створено демонстраційний додаток, в якому відображається робота розробленого модулю, та надається можливість зберігати результати виконання в текстовий файл.

Користувач має змогу власноруч задавати усі параметри моделювання та задавати положення контрольних точок.

Дана система істотно спрощує процес геометричного моделювання та дозволяє досягти чіткого результату на мобільному пристрої.

Користувач має змогу створювати початкову поверхню за допомогою виконуваного файлу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Голованов Н.Н. Геометрическое моделирование. — М.: Издательство Физико-математической литературы, 2002. — 472 с. — ISBN 5-94052-048-0.
2. Гильберт Д., Кон-Фоссен С. Наглядная геометрия. — М.: Наука, 1981.
3. Бачишин Б. Д. Дослідження точності трансформації растру / Б. Д. Бачишин, Р. Б. Шульган, О. Є. Янчук // Вісник Національного університету водного господарства та природокористування: зб. наук. пр. – Рівне, 2006. – Вип. 1 (33). – С. 179–185.
4. Карпінський Ю. О. Афінне трансформування координат методом скінченних елементів / Ю. О. Карпінський // Вісник геодезії та картографії. – 2002. – № 4. – С. 23–27.
5. Кондратюк О. В. Координатне інтегрування різнорідних картографічних матеріалів у ГІСКОРДОН / О. В. Кондратюк // Вісник геодезії та картографії. – 2010. – № 6 (69). – С. 33–38.
6. Орел Д.С. Програмне забезпечення для взаємодії PDM та CAD систем / Д.С. Орел, Л.О. Левченко // XVI міжнародна науково-практична конференція аспірантів, магістрантів, студентів «Сучасні проблему наукового забезпечення енергетики», 24-27 квітня, 2018 року. – м. Київ, 2018. – С. 298.
7. Ivanov D. & Sokolov B., (2010), Adaptive Supply Chain Management. — L.: Springer. — 200 p.
8. Абросимова А. Технологии и люди. Сложности внедрения ERP-систем / Анна Абросимова // Управление человеческим потенциалом. – 2006. – №1. – С. 25–30.
9. Ловыгин А. А., Васильев А. В. Современный станок с ЧПУ и CAD/CAM система / А. А. Ловыгин, А.В. Васильев. — Издательство Эльф ИПР, 2006 г. – 240 с.
10. Втюрин В.А. Автоматизированные системы управления технологическими процессами. Основы АСУТП. / В.А. Втюрин. — Москва: «Солон-Пресс». – 2006г. – 124 с.

11. Панкова Л. А. Способы создания универсального инструментария для компьютерного моделирования / Л.А. Панкова, В.А. Пронина. // Проблемы управления (Control Sciences). – 2006. – № 6. – С. 2-5.
12. Пилюгин В. В. Геометрическое моделирование / В. В. Пилюгин, Л. Н. Сумароков // Матем. моделирование, 6:5, 1994. С 21–36.
13. Райковська Г. Геометричне моделювання – основа конструкторськотехнологічних здібностей / Г. Райковська, В. Головня // Нова пед. думка : наук.- метод. журн. - 2013. - № 1 ч. 2. - С. 68-70.
14. Хейфец А.Л. Инженерная 3d-компьютерная графика: учеб. пособие для бакалавров / А.Л. Хейфец, А.Н. Логиновский, И.В. Буторина, В.Н. Васильева; под ред. А.Л. Хейфеца – 2-е изд., перераб. и доп. – М.: Юрайт, 2014. – 464с.

ДОДАТОК А

Динамічне середовище геометричного моделювання для мобільних телефонів

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5156_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ ТР51112_18Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ ТР51112_18Б 12-1	CreateMesh.cs CreateVertices.cs ControlCamera.cs	Основні компоненти
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ ТВ41112_18Б 13-1	Додаток В.doc	Опис програмного модуля

ДОДАТОК Б

Динамічне середовище геометричного моделювання для мобільних телефонів.

Текст програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5156_19Б 12-1

Аркушів6

Київ 2019

```
//CreateMesh
   using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;

    public class CreateMesh : MonoBehaviour
    {
        private Point3D point00;
        private Point3D point01;
        private Point3D point02;
        private Point3D point10;
        private Point3D point11;
        private Point3D point12;
        private Point3D point20;
        private Point3D point21;
        private Point3D point22;

        public void setPoint00(Point3D point00)
        {
            this.point00 = point00;
        }
        public void setPoint01(Point3D point01)
        {
            this.point01 = point01;
        }
        public void setPoint02(Point3D point02)
        {
            this.point02 = point02;
        }
        public void setPoint10(Point3D point10)
        {
            this.point10 = point10;
        }
        public void setPoint11(Point3D point11)
        {
            this.point11 = point11;
        }
        public void setPoint12(Point3D point12)
        {
            this.point12 = point12;
        }
        public void setPoint20(Point3D point20)
        {
            this.point20 = point20;
        }
        public void setPoint21(Point3D point21)
        {
            this.point21 = point21;
        }
        public void setPoint22(Point3D point22)
        {
            this.point22 = point22;
```

```
    }
    public Point3D getPoint00()
    {
        return point00;
    }
    public Point3D getPoint01()
    {
        return point01;
    }
    public Point3D getPoint02()
    {
        return point02;
    }
    public Point3D getPoint10()
    {
        return point10;
    }
    public Point3D getPoint11()
    {
        return point11;
    }
    public Point3D getPoint12()
    {
        return point12;
    }
    public Point3D getPoint20()
    {
        return point20;
    }
    public Point3D getPoint21()
    {
        return point21;
    }
    public Point3D getPoint22()
    {
        return point22;
    }

    public Point3D[] getPointMas()
    {
        Point3D[] result = new Point3D[9];
        result[0] = point00;
        result[1] = point01;
        result[2] = point02;
        result[3] = point10;
        result[4] = point11;
        result[5] = point12;
        result[6] = point20;
        result[7] = point21;
        result[8] = point22;
        return result;
}
```

```
public void setPointMas(int i, Vector3 vector)
{
    Point3D point = new Point3D(vector.x * 1000, vector.y * 1000, vector.z * 1000);

    switch (i)
    {
        case 0:
            this.point00 = point;
            break;
        case 1:
            this.point01 = point;
            break;
        case 2:
            this.point02 = point;
            break;
        case 3:
            this.point10 = point;
            break;
        case 4:
            this.point11 = point;
            break;
        case 5:
            this.point12 = point;
            break;
        case 6:
            this.point20 = point;
            break;
        case 7:
            this.point21 = point;
            break;
        case 8:
            this.point22 = point;
            break;

    }
}


// Start is called before the first frame update
void Start()
{
    this.GetComponent<LoadFromFile>().fileName = "SurfaceDefault.txt";
    this.GetComponent<LoadFromFile>().load();
    //point00 = new Point3D(450, 450, 450);
    //point01 = new Point3D(250, 450, 350);
    //point02 = new Point3D(400, 600, 200);
    //point10 = new Point3D(600, 500, 700);
    //point11 = new Point3D(600, 900, 700);
    //point12 = new Point3D(800, 800, 700);
    //point20 = new Point3D(900, 300, 1000);
    //point21 = new Point3D(900, 600, 1000);
```

```
    //point22 = new Point3D(1200, 500, 1000);
    //createMesh();
}

public class Point3D
{
    private float x;
    private float y;
    private float z;

    public float getX()
    {
        return x;
    }
    public float getY()
    {
        return y;
    }
    public float getZ()
    {
        return z;
    }
    public void setX(float x)
    {
        this.x = x;
    }
    public void setY(float y)
    {
        this.y = y;
    }
    public void setZ(float z)
    {
        this.z = z;
    }

    public Point3D(float x, float y, float z)
    {
        this.x = x/10;
        this.y = y/10;
        this.z = z/10;
    }


    public override string ToString()
    {
        return x *10 + " " + y*10 + " " + z*10;
    }
}

public void createMesh()
{
    this.GetComponent<MeshFilter>().mesh.vertices = createVertices3D();
```

```
        this.GetComponent<MeshFilter>().mesh.normals = createNormals();
        this.GetComponent<MeshFilter>().mesh.uv = createUV();
    }

    private Vector3[] createVertices3D()
    {
        Dictionary<int, List<Point3D>> verticalPoint = constructionDictionaryVerticalLine();

        List<Point3D> verticalPointList = new List<Point3D>();

        foreach (KeyValuePair<int, List<Point3D>> keyValue in verticalPoint)
        {

            foreach (Point3D point in keyValue.Value)
            {
                verticalPointList.Add(point);
            }
        }


        Vector3[] vertices = new Vector3[verticalPointList.Count];
        int count = 0;
        foreach (Point3D point in verticalPointList)
        {
            vertices[count++] = new Vector3(point.getX(), point.getY(), point.getZ());
        }
        return vertices;
    }

    private Dictionary<int, List<Point3D>> constructionDictionaryVerticalLine()
    {
        Point3D tempP0;
        Point3D tempP1;
        Point3D tempP2;
        Dictionary<int, List<Point3D>> verticalLine = new Dictionary<int, List<Point3D>>();

        int count = 0;

        for (double v = 0; v <= 1; v += 0.1)
        {
            tempP0 = calculatePoint3D(point00, point01, point02, (float)v);
            tempP1 = calculatePoint3D(point10, point11, point12, (float)v);
            tempP2 = calculatePoint3D(point20, point21, point22, (float)v);

            verticalLine.Add(count++, constructionPointVerticalLine(tempP0, tempP1, tempP2));
        }

        return verticalLine;
    }

    private List<Point3D> constructionPointVerticalLine(Point3D tempP0, Point3D tempP1,
Point3D tempP2)
```

```
    {
        List<Point3D> pointList = new List<Point3D>();

        for (double u = 0; u <= 1; u += 0.1)
        {
            Point3D point = calculatePoint3D(tempP0, tempP1, tempP2, (float)u);
            pointList.Add(point);
        }

        return pointList;
    }

    private Point3D calculatePoint3D(Point3D p0, Point3D p1, Point3D p2, float v)
    {
        return new Point3D(
                coordinateCalculation(p0.getX(), p1.getX(), p2.getX(), v),
                coordinateCalculation(p0.getY(), p1.getY(), p2.getY(), v),
                coordinateCalculation(p0.getZ(), p1.getZ(), p2.getZ(), v)
        );
    }


    private float coordinateCalculation(float coord0, float coord1, float coord2, float t)
    {
        return coord0 * (1 - t) * (1 - t) + coord1 * 2 * t * (1 - t) + coord2 * t * t;
    }

    private Vector3[] createNormals()
    {
        Vector3[] normals = new Vector3[121];
        for (int i = 0; i < 121; i++)
            normals[i] = -Vector3.forward;
        return normals;
    }


    private Vector2[] createUV()
    {
      Vector2[] uv = new Vector2[121];
      for (int i = 0; i < 119; i += 4)
     {
         uv[i] = new Vector2(0, 0);
        uv[i + 1] = new Vector2(1, 0);
        uv[i + 2] = new Vector2(0, 1);
        uv[i + 3] = new Vector2(1, 1);
      }
     return uv;
    }
}
```

ДОДАТОК В

Динамічне середовище геометричного моделювання для мобільних телефонів.

Опис програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5156_19Б 13-1

Аркушів6

Київ 2019

АНОТАЦІЯ

Розділ містить опис частини, яка слугує для побудови поверхні, яка також являється структурною одиницею програмного продукту, та забезпечує поєднання можливостей усіх інших модулів для виконання поставлених перед системою завдань. Призначенням цієї частини є побудова поверхні Безьє. Модуль надає можливість отримувати необхідні дані про опорні точки поверхні , оновлювати та будувати поверхню. Модуль написано мовою програмування C#, з використанням технологій Unity3D.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ 4
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 5
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ 6
4. ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ 7
5. ВИКЛИК І ЗАВАНТАЖЕННЯ 8
6. ВХІДНІ ТА ВИХІДНІ ДАНІ 9

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглядається один з програмних модулів системи — модуль для побудови поверхні Безьє з кодом УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5156_19Б 12-1, що міститься у файлі CreateMesh.cs. Модуль призначений для управління підсистемами, які відповідають за геометричне моделювання поверхонь, які зберігаються в текстовому файлі. Користувач має можливість створювати новий файл, редагувати або видаляти необхідні дані в файлі.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Призначенням модулю для побудови поверхні є забезпечення побудови поверхні Безьє на інтерфейсній частини та безпосереднього прийняття даних з модуля для роботи з файловою системою а також обмін інформації із інтерфейсом користувача. Використання такого шаблону дозволяє створювати програмне забезпечення, де інтерфейс і логіка роботи модуля для побудови є незалежними компонентами, що дає можливість використовувати його для зменшення навантаження на клієнтську частину системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Слідкувати за зміною інформації про підсистеми є головним завданням модуля. При запуску системи модуль дістає всю інформацію, яка міститься в файлі, для відображення даних на користувацькому інтерфейсі. Також модуль оброблює інформацію, надаючи змогу додавати запис, змінювати або видаляти необхідний.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модуль розроблено у середовищі розробки Microsoft Visual Studio 2017, що забезпечує набір сервісних функцій та діалог з розробником, на комп’ютері, що використовував операційну систему Windows 7. З файловою системою, комп’ютер з’єднувався за допомогою бібліотеки System.IO.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний модуль реалізований як окремий клас, який забезпечує існування клієнтської частини та бізнес-логіки окремо від одного, але разом із цим запуск обох компонентів відбувається одночасно.

Для використання даного модулю не потрібно ніяких дій, оскільки він автоматично спрацьовує після запуску клієнтського додатку.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є інформація, з текстового файлу.

Вихідними даними програмного модуля є дані, за допомогою яких Unity3D будує поверхню.